Gedächtnisprotokoll TI2 (SoSe 2023)

Total Punkte : 120

Aufgabe 1: Interrupts und Systemaufrufe

- ein Vorteil von Interrupts gegenüber Polling
- ein Vorteil von Polling gegenüber Interrupts
- zwei Gründe, warum Systemaufrufe notwendig sind
- Wie werden Systemaufrufe implementiert? Dabei auf den Wechsel vom Usermode (Ring 3) in den Kernelmode (Ring 0) eingehen.
- Wie implementiert ein C-Entwickler im POSIX-Standard in der Regel Systemaufrufe?

Aufgabe 2: Prozesse und Threads

- Nenne zwei Vorteile, die Threads gegenüber Prozessen haben.
- Bestandteile von PCB
- Kann ein Betriebssystem Prozesse erstellen / mit diesen arbeiten, wenn es keine Memory Management Unit hat? (MMU = Hardware, die für Übersetzung von virtuellen Adressen in physikalische verantwortlich ist.)
- Kann ein Betriebssystem Threads erstellen / mit diesen arbeiten, wenn es keine Memory Management Unit hat?

Aufgabe 3: Deadlocks

- Nenne die vier notwendigen Bedingungen für einen Deadlock und beschreibe sie kurz.
- Philosophen Problem gegeben: vier Philosophen wollen nach dem linken Stäbchen greifen -> Deadlock.
- Wie kann der Deadlock vermieden werden? (Zusätzlicher Zustand, Asymmetrie)
- Welche Bedingung für einen Deadlock wird dadurch nicht mehr erfüllt?

Aufgabe 4: Scheduling (17 Punkte)

- Folgende Tabelle mit Prozessen A-E, sowie deren Ankunftszeit und Ausführungsdauer gegeben. Es soll in eine Tabelle eingetragen werden, zu welcher Zeiteinheit (0 - 14) welcher Prozess läuft. Der Scheduling-Algorithmus ist Shortest Process Next (SPN)

Prozess	Ankunftszeit	Ausführungsdauer
A	0	3
B	2	4
C	4	4
D	6	3
E	8	1

- Nenne einen Vorteil vom SPN-Algorithmus

- Nenne einen Nachteil vom SPN-Algorithmus
- Nenne einen Scheduling-Algorithmus für Echtzeitprozesse
- Was bedeutet “präemptiv” im Kontext von Scheduling?
- Angenommen man hat ein System mit einem interaktiven Prozess (Prozess, der zwischendurch immer wieder auf Eingaben des Benutzers wartet), sowie mehrere CPU- und E/A-lastige Hintergrundprozesse. Wie kann man dafür sorgen, dass der interaktive Prozess nicht allzu lange auf Ausführung warten muss und möglichst ohne große Wartezeit ausgeführt wird.

Aufgabe 5: Virtueller Speicher

- Größe des virtuellen Speichers(?) des Prozesses angeben (Seitentabelle von 0 bis 7, 4 Einträge waren besetzt; Seitengröße = 1 KiB)
- Resident Set Size des Prozesses bei der Initialisierung angeben (Seitentabelle von 0 bis 7, 4 Einträge waren besetzt; Seitengröße = 1 KiB)
- 4 virtuelle Adressen in reale Adressen umrechnen (Paging)
- Was passiert bei folgender Adresse? (Es kam zum page fault)
- Was ist ein page fault?

Aufgabe 6: Cache

- Fast alle modernen CPUs verwenden einen Cache. Was sind die Vorteile von einem Cache?
- Warum verzichten alte CPUs auf Cache?
- 2 Varianten eines Programms wie Probeklausur. welche hat höhere Cacheeffizienz? Begründen Sie:
  - Variante A

```
void do_calculation(int *inout, int length) {
    for(unsigned i = 0; i < length; i++) {
        inout[i] = sqrt(inout[i]);
    }
    for(unsigned i = 0; i < length; i++) {
        inout[i] = inout[i] * 1.337;
    }
}
```

  - Variante B

```
void do_calculation(int *inout, int length) {
    for(unsigned i = 0; i < length; i++) {
        inout[i] = sqrt(inout[i]);
        inout[i] = inout[i] * 1.337;
    }
}
```

Aufgabe 7: Internet

- TCP/IP-Schichtenmodell: Erkläre die Schichten Application-Layer, Transport-Layer, Internet-Layer, Host-to-Network-Layer und nenne je ein Beispielprotokoll
- Erkläre das Designprinzip “Best Effort” des Internets

- Manchmal gehen IP-Pakete verloren. Nenne eine Möglichkeit, wie man dennoch sicherstellen kann, dass ein Paket beim Empfänger ankommt

Aufgabe 8: Programmieraufgabe (14 Punkte)

- Schleife gegeben, die für die Zahlen 0 bis einschließlich 17 schauen sollte, ob diese durch 2 und 3, nur durch 2, nur durch 3 oder durch keine der beiden Zahlen teilbar war. Dann sollte entsprechend “foobar”, “foo”, “bar” oder “-” ausgegeben werden. Es mussten vier Felder ergänzt werden:
  - Einmal die drei if-Bedingungen, also `i % 2 == 0 && i % 3 == 0`, `i % 2 == 0`, sowie `i % 3 == 0`
  - Einmal das `puts("-");`
- Es war C-Code gegeben, der einen Array mit der Größe SIZE=100 mit dem Wert 1 initialisiert, sowie Element 0 und 1 auf 0 gesetzt. Anschließend wurde folgende Schleife ausgeführt: Es wurde über alle Elemente iteriert (`for(unsigned i = 0; i < SIZE; i++) {...}`). Befindet sich im Array an dieser Stelle der Wert 0, so wird mit dem nächsten Element fortgefahren. Andersfalls wird eine weitere Schleife ausgeführt, die über alle Vielfachen der Zahl iteriert (`for(unsigned j = (i << 1); j < SIZE; j += i) {...}`) und im Array an diesen Stellen den Wert auf 0 setzt. Zum Schluss, nachdem diese Schleifen beendet sind, werden alle Zahlen, bei denen der Wert nicht 0 ist, ausgegeben.
  - Es sollte erklärt werden, was dieses Programm für eine Ausgabe hat: Es gibt die Primzahlen von 0 bis SIZE-1 aus

Aufgabe 9: Testatfragen (10 Punkte)

- Ziele von BS
- Ziele von Scheduling
- Erlauben es User-Level-Threads anwendungsspezifischer zu optimieren